Branching Processes of Conservative Nested Petri Nets*

Daniil Frumin and Irina Lomazova

National Research University Higher School of Economics (HSE),
difrumin@edu.hse.ru, ilomazova@hse.ru

Abstract

Nested Petri nets (NP-nets) is an extension of the Petri nets formalism within the "nets-within-nets" approach, when tokens in a marking are themselves Petri nets which have autonomous behavior and synchronize with the system net. The formalism of NP-nets allows modeling multi-level multi-agent systems with a dynamic structure in a natural way. In this paper we define branching processes and unfoldings for conservative NP-nets, i.e., NP-nets with a persistent set of agents. We show that NP-nets unfoldings satisfy the fundamental property of unfoldings, and thus can be used for verification of conservative NP-nets in line with classical unfolding methods.

1 Introduction

Nested Petri nets (NP-nets) [20, 22] is an extension of high-level Petri nets according to the "nets-within-nets" approach. Nets-within-nets have been extensively studied in the Petri net literature, as an approach for modeling active objects, mobility and dynamics in distributed systems [2, 12, 18, 28].

NP-nets are a convenient formalism for modeling systems of dynamic interacting agents: each agent is represented by a net token, and the agents are distributed in a system net. Levels in NP-nets are coordinated via synchronized transitions (simultaneous firing of transitions in adjacent levels of the model). Because of the loosely-coupled multilevel structure, NP-nets can be used for the effective modeling of mutli-agent systems (see [4, 10, 23, 21, 3] for examples).

Unlike other nets-within-nets formalisms (such as Elementary Object Systems or Hypernets [24, 17]), the behavior of an NP-net is defined according to the value semantics, whereby each net token represents an independent object. Due to this, NP-nets inherit useful properties (such as locality and monotonicity) of classical Petri nets, e.g., NP-nets allow for a conservative extension to the Linear Logic calculus of Girard [8]. Boundedness and liveness (under some restrictions) can be deduced for NP-nets in a compositional way, i.e., from boundedness, respectively liveness, of nested net components [4]. Note also that NP-nets are strictly more expressive than Petri nets, but are not Turing-complete [20].

In this paper we consider conservative NP-nets, in which agents (net tokens) cannot be created or destroyed, but can be moved from one location to another and can change their inner states. Even with those restrictions, NP-nets can be employed for modeling interesting systems. In our case, we are keeping track of the individual net tokens, during the unfolding process, which is the prime reason for us considering only conservative NP-nets at this point.

Multi-agent systems are usually highly concurrent. A crucial problem in verification of highly concurrent systems is a large number of interleavings — possible sequencings of events in the system. This leads to what is usually referred to as a state space explosion problem, when a set of all reachable states in a transition system grows vastly upon adding a small component to the system.

*This study was carried out within the National Research University Higher School of Economics' Academic Fund.

To tackle this problem, various methods based on non-interleaving semantics have been proposed and studied. One of the most popular methods is based on generating complete finite prefixes of *unfoldings*. The unfoldings theory was originally developed by Winskel, Nielsen, and Plotkin [27]. McMillan [25] was the first to use unfoldings for verification. He introduced the concept of *complete finite prefixes* of unfoldings and demonstrated the applicability of this approach to the verification of asynchronous circuits.

The original McMillan's algorithm was used to solve the *executability* problem, i.e., to check whether a given transition can fire in the net. This algorithm also can be used for checking whether a net is deadlock-free and for solving some other problems. Later, numerous improvements to the algorithm have been proposed ([7, 26, 11] to name a few); and the approach has been applied to other models of computation, such as process algebras [19] and high-level Petri nets [15].

The general method for truncating unfoldings, which abstracts from the information one wants to preserve in the finite prefix of the unfolding, was proposed in [16, 14]. This method is based on the notion of a *cutting context*, and can be transferred to our definition of branching processes and unfoldings of conservative NP-nets.

The paper is organized as follows. In the Section 2 we present the basic notions of Petri nets, branching processes and unfoldings. In Section 3 we define nested Petri nets (NP-nets) and examine a particular class of NP-nets – conservative NP-nets. In Section 4, we study a compositional construction on an NP-net: specifically, an NP-net in which each individual component has been unfolded. In Section 5 we present the main contribution of our paper: the definition of a branching process of a conservative NP-net. Lastly, we discuss the applicability of our construction to the verification algorithms based on the canonical prefixes of unfoldings.

2 Preliminaries

Multisets. Let S be a finite set. A *multiset* m over a set S is a function $m : S \to Nat$, where Nat is the set of natural numbers (including zero), in other words, a multiset may contain several copies of the same element.

For two multisets m, m' we write $m \subseteq m'$ iff $\forall s \in S : m(s) \leqslant m'(s)$ (the inclusion relation). The sum and the union of two multisets m and m' are defined as usual: $\forall s \in S : (m + m')(s) = m(s) + m'(s)$, $(m \cup m')(s) = max(m(s), m'(s))$.

P/T-nets. Let P and T be two finite disjoint sets of *places* and *transitions* and let $F \subseteq (P \times T) \cup (T \times P)$ be a *flow relation*. Then $N = (P, T, F)$ is called a *P/T-net*.

A *marking* in a P/T-net $N = (P, T, F)$ is a multiset over the set of places P. By $\mathcal{M}(N)$ we denote a set of all markings in N. A *marked P/T-net* (N, M_0) is a P/T-net together with its *initial marking* M_0.

Pictorially, P-elements are represented by circles, T-elements by boxes, and the flow relation F by directed arcs. Places may carry tokens represented by filled circles. A current marking m is designated by putting $m(p)$ tokens into each place $p \in P$.

For a transition $t \in T$, an arc (x, t) is called an *input arc*, and an arc (t, x) — an *output arc*. For each node $x \in P \cup T$, we define the *pre-set* as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and the post-set as $x^\bullet = \{y \mid (x, y) \in F\}$.

We say that a transition t in P/T-net $N = (P, T, F)$ is *enabled* at a marking M if ${}^\bullet t \subseteq M$. An enabled transition may *fire*, yielding a new marking $M' = M - {}^\bullet t + t^\bullet$ (denoted $M \xrightarrow{t} M'$). A marking M is called *reachable* if there exists a (possibly empty) sequence of firings $M_0 \xrightarrow{t_1}$

$M_1 \xrightarrow{t_2} M_2 \to \cdots \to M$ from the initial marking to M. By $\mathcal{RM}(N)$ we denote the set of all reachable markings in N.

A marking M is called *safe* iff for all places $p \in P$ we have $M(p) \leqslant 1$. A marked P/T-net N is called *safe* iff every reachable marking $M \in \mathcal{RM}(N)$ is safe. A *reachability graph* of a P/T-net (N, M_0) presents detailed information about the net behavior. It is a labeled directed graph, where vertices are reachable markings in (N, M_0), and an arc labeled by a transition t leads from a vertex v, corresponding to a marking M, to a vertex v', corresponding to a marking M' iff $M \xrightarrow{t} M'$ in N.

Branching processes and unfoldings of P/T-nets. Unfoldings are used to define non-sequential (true concurrent) semantics of P/T-nets, and complete prefixes of unfoldings are used for verification. Here we give necessary basic notions and definitions, connected with unfoldings. Further details can be found in [6, 5].

Let $N = (P, T, F)$ be a P/T-net. The following relations are defined on the set $P \cup T$ of nodes in N:

1. the *causality* relation, denoted as $<$, is the transitive closure of F, and $\leqslant$ is the reflexive closure of $<$; if $x < y$, we say that y causally depends on x.
2. the *conflict* relation, denoted as $\#$: for nodes $x, y \in P \cup T$, $x\#y := \exists t, t' \in T.t \neq t' \wedge {}^\bullet t \cap {}^\bullet t' \neq \varnothing \wedge t \leqslant x \wedge t' \leqslant y$;
3. the *concurrency* relation, denoted as *co*: two nodes are *concurrent* if they are not in conflict and neither of them causally depends on the other.

For a set B of nodes we write $co(B)$ iff all nodes in B are pairwise concurrent.

An *occurrence net* is a safe P/T-net $ON = (B, E, G)$ s.t.

1. ON is acyclic;
2. $\forall p \in B\colon |{}^\bullet p| \leqslant 1$;
3. $\forall x \in B \cup E$ the set $\{y \mid y < x\}$ is finite, i.e., each node in ON has a finite set of predecessors;
4. $\forall x \in B \cup E\colon \neg(x\#x)$, i.e., no node is in self-conflict.

In occurrence nets, elements from B are usually called *conditions* and elements from E are called *events*.

A *configuration* C in an occurrence net $ON = (B, E, G)$ is a non-conflicting subset of nodes, which is downwards-closed under $<$, i.e., $\forall x, y \in C\colon \neg(x\#y)$, and $(x < y) \wedge y \in C$ implies $x \in C$. For each $x \in B \cup E$ we define a *local configuration of* x to be $[x] = \{y \mid y \in B \cup E, y < x\}$. The definition of a local configuration can be straightforwardly generalized to any non-conflicting set of nodes $X \subseteq B \cup E$, namely $[X] = \{y \mid y \in B \cup E, x \in X, y < x\}$.

We define the set of branching processes of a given marked P/T-net $N = (P, T, F, M_0)$ using the so-called *canonical representation*.

The set $\mathcal{C}$ of *canonical names* for N is defined recursively to be the smallest set s.t. if $x \in P \cup T$ and A is a finite subset of $\mathcal{C}$, then $(A, x) \in \mathcal{C}$.

A $\mathcal{C}$-Petri net is an occurrence net (B, E, G) such that:

- $B \cup E \subseteq \mathcal{C}$;
- $\forall (A, x) \in B \cup E$, ${}^\bullet(A, x) = A$.

The initial marking of a $\mathcal{C}$-Petri net is a subset of nodes $\{(\varnothing, x) \mid (\varnothing, x) \in B\}$. For each $\mathcal{C}$-Petri net CN, the morphism h maps the nodes of CN to the nodes of N: $h((A, x)) = x$. If $h(y) = z$, we say that y is labeled by z.

Let S be a (finite or infinite) set of $\mathcal{C}$-Petri nets. The union of S is defined component-wise, i.e., $\bigcup S = (\bigcup_{(P,T,F,M) \in S} P, \bigcup_{(P,T,F,M) \in S} T, \bigcup_{(P,T,F,M) \in S} F, \bigcup_{(P,T,F,M) \in S} M)$.

The set of *branching processes* of a marked P/T-net $N = (P, T, F, M_0)$ is defined as the smallest set satisfying the following conditions:

1. The occurrence net $(I, \varnothing, \varnothing)$, where $I = \{(\varnothing, p) \mid p \in M_0\}$ (consisting of conditions I and having no events), is a branching process.
2. Let $\mathcal{B}_1$ be a branching process and M be a reachable marking of $\mathcal{B}_1$, and $M' \subseteq M$, such that $h(M') = {}^\bullet t$ for some t in T. Let $\mathcal{B}_2$ be a net obtained by adding an event (M', t) and conditions $\{(\{(M', t)\}, p) \mid p \in t^\bullet\}$ to $\mathcal{B}_1$. Then $\mathcal{B}_2$ is a branching process.
3. Let $\mathcal{BB}$ be a (finite, or infinite) set of branching processes. The union $\bigcup \mathcal{BB}$ is a branching process.

An example of a P/T-net and its branching process is shown in Figs. 1 and 2. The P/T-net PN_1 has the initial marking $\{p_1\}$ and is shown in Fig. 1. One of its possible branching processes is shown in Fig. 2, in which the labeling function h is indicated by labels on nodes.

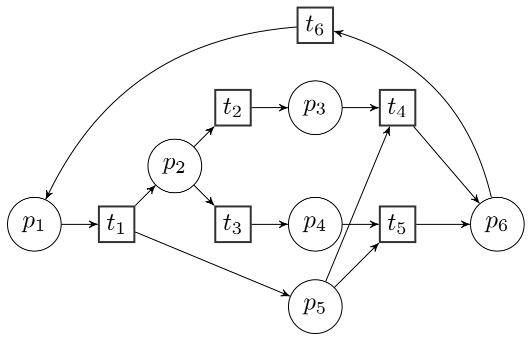

Figure 1: Petri net PN_1

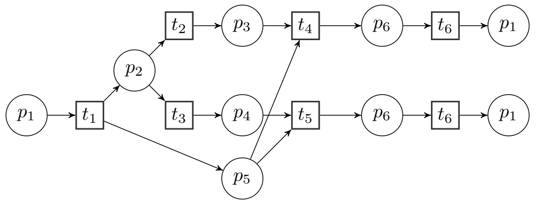

Figure 2: Branching process of PN_1

A branching process $\mathcal{B}_1 = ((P_1, E_1, F_1), h_1)$ is called a *prefix* of a branching process $\mathcal{B}_2 = ((P_2, E_2, F_2), h_2)$ (denoted $\mathcal{B}_1 \sqsubseteq \mathcal{B}_2$) iff $P_1 \subseteq P_2$ and $E_1 \subseteq E_2$.

The maximal branching process of a net N w.r.t the prefix relation $\sqsubseteq$ is called the *unfolding* of N and is denoted by $U(N)$.

The *fundamental property of P/T-nets unfoldings* [5] states that the behavior of the unfolding is equivalent to the behavior of the original net. Formally it can be formulated as follows.

Fundamental property of P/T-nets unfoldings. Let M be a reachable marking in a P/T- net N, and let M_U be a reachable marking in $U(N)$ s.t. $h(M_U) = M$. Then

1. if there is a step $M_U \xrightarrow{t_U} M'_U$ of $U(N)$, then there is a step $M \xrightarrow{t} M'$ of N, such that $h(t_U) = t \wedge h(M'_U) = M'$;
2. if there is a step $M \xrightarrow{t} M'$ of N, then there is a step $M_U \xrightarrow{t_U} M'_U$ in $U(N)$, such that $h(t_U) = t \wedge h(M'_U) = M'$.

In other words, the fundamental property of unfoldings states that the reachability graph of the unfolding is isomorphic to the reachability graph of the P/T-net. This property is critical for the use of unfoldings in semantic study and verification. Unfoldings were defined and studied for different classes of Petri nets, namely for high-level Petri nets [15], contextual nets [1], time

Petri nets [9], Hypernets [24] (to name a few). All these constructions has similar properties, which act as a "sanity check".

Further in the paper we define an unfolding operation for nested Petri nets, which posses a similar fundamental property.

3 Nested Petri Nets

In this paper we deal with nested Petri nets (NP-nets) — in particular, a proper subclass of NP-nets called *strictly conservative* NP-nets. The basic definition of nested Petri nets can be found in [20, 23]. Here we give a reduced definition, sufficient for defining conservative NP-nets.

In *nested Petri nets (NP-nets)*, tokens may be Petri nets themselves. An NP-net consists of a *system net* and *element nets*. We call these nets the NP-net *components*. Marked element nets are *net tokens*. Net tokens, as well as usual black dot tokens, may reside in places of the system net. Some transitions in NP-net components may be labeled with *synchronization labels*. Unlabeled transitions in NP-net components may fire autonomously, according to the usual rules for Petri nets. Labeled transitions in the system net should synchronize with transitions (labeled by the same label) in net tokens involved in this transition firing.

In strictly conservative NP-nets, net tokens cannot evolve or disappear. They can "move" from one place in a system net to another and "change" their marking, i.e., inner state. In the basic NP-net formalism new net tokens may be created, copied and removed as usual Petri net tokens. It should be noted that although this restriction is rather strong, many interesting multi-agent systems can be modeled with conservative NP-nets.

Here we consider *safe* and *typed* NP-nets, i.e., each place in a system net can contain no more than one token: either a black dot token, or a net token of a specific type.

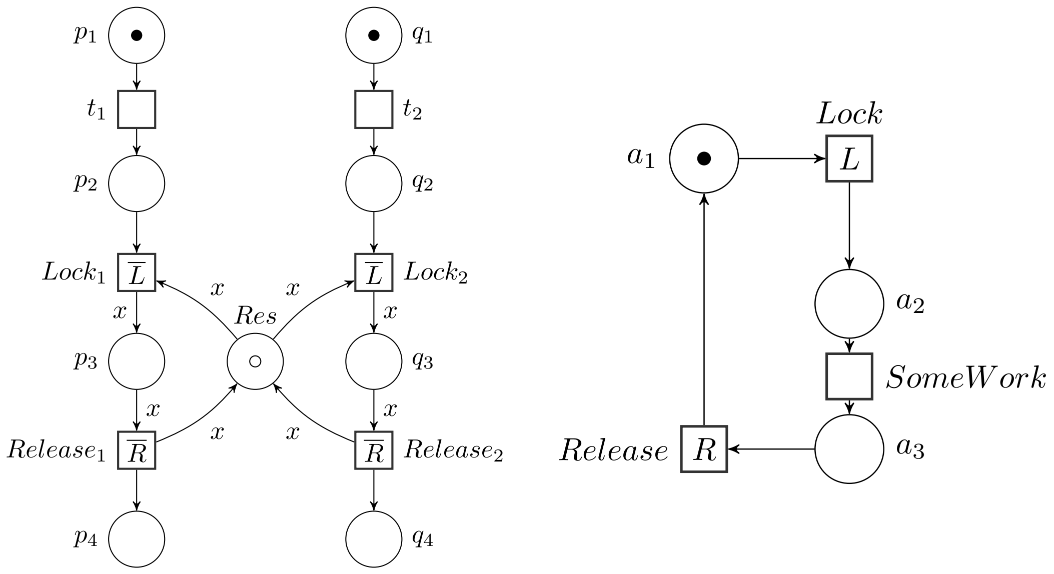

Figure 3: NP-net NP_1

Figure 3 provides an example of a nested Petri net NP_1. On the left one can see a system net. The token residing in the place *Res* is a net token. Its structure and initial marking is shown on the right side of the figure. The net token represents some sort of resource (for example, a networking or a computational one), capable of performing some internal work (actions). Two threads are trying to access the same resource, but the locking mechanism is preventing them from accessing it simultaneously. The system net synchronizes with the element nets via transitions $Lock_1$, $Lock_2$ and $Release_1$,$Release_2$.

Definition 1 (Nested Petri nets). *Let Type be a set of types, Var — a set of typed (over Type)* variables*, and Lab — a set of labels. A (typed)* nested Petri net (NP-net) *NP is a tuple* $(SN, (EN_1, \ldots, EN_k), \upsilon, \lambda, W)$*, where*

- $SN = (P_{SN}, T_{SN}, F_{SN})$ *is a P/T net called a* system net*;*
- *for* $i = \overline{1,k}$*,* $EN_i = (P_{EN_i}, T_{EN_i}, F_{EN_i})$ *is a P/T net called an* element net*, where all sets of places and transitions in the system and element nets are pairwise disjoint; we suppose, each element net is assigned a type from Type;*
- $\upsilon : P_{SN} \to Type \cup \{\bullet\}$ *is a* place-typing function*;*
- $\lambda : T_{NP} \to Lab$ *is a partial transition labeling function, where* $T_{NP} = T_{SN} \cup T_{EN_1} \cup \cdots \cup T_{EN_k}$*; we write that* $\lambda(t) = \perp$ *when* λ *is undefined at* t*.*
- $W : F_{SN} \to Var \cup \{\bullet\}$ *is an* arc labeling function *s.t. for an arc* r *adjacent to a place* p *the type of* $W(r)$ *coincides with the type of* p*.*

A marked element net is called a net token.

In what follows for a given NP-net by $A_{net} = \{(EN, m) \mid \exists i = 1, \ldots, k : EN = EN_i, m \in \mathcal{M}(EN_i)\}$ we denote the set of all (possible) net tokens, and by $A = A_{net} \cup \{\bullet\}$ the set of all net tokens extended with a black dot token.

Now we come to defining NP-net behavior.

A *marking* M in an NP-net NP is a function mapping each $p \in P_{SN}$ to some (possibly empty) multiset $M(p)$ over A. Thus a marking in an NP-net is defined as a marking of its system net. By abuse of notation, a set of all markings of an NP-net NP will be denoted by $\mathcal{M}(NP)$. We say that a net token (EN, m) resides in p (under marking M), if $M(p) = \{(EN, m)\}$.

Let t be a transition in SN, ${}^\bullet t = \{p_1, \ldots, p_i\}$, $t^\bullet = \{q_1, \ldots, q_j\}$ be sets of its pre- and post-elements. Then $W(t) = \{W(p_1, t), \ldots, W(p_i, t), W(t, q_1), \ldots, W(t, q_j)\}$ will denote a set of all variables in arc labels adjacent to t. A *binding* of t is a function b assigning a value $b(v)$ (of the corresponding type) from A to each variable v occurring in $W(t)$.

A transition t in SN is *enabled* in a marking M w.r.t. a binding b iff $\forall p \in {}^\bullet t : W(p, t)(b) \subseteq M(p)$, i. e. each input place p adjacent to t contains a value of input arc label $W(p, t)$.

The enabled transition *fires* yielding a new marking M', write $M \to M'$, such that for all places p, $M'(p) = (M(p) \backslash W(p, t)(b)) \cup W(t, p)(b)$.

For net tokens from A_{net}, which serve as values for input arc variables from $W(t)$, we say, that they are *involved* in the firing of t. (They are removed from input places and brought to output places of t).

There are three kinds of steps in an NP-net NP.

An element-autonomous step. Let t be a transition without synchronization labels in a net token. Then an autonomous step is a firing of t according to the usual rules for P/T-nets. An autonomous step in a net token does not change the residence of this net token.

A *system-autonomous step* is the firing of an unlabeled transition $t \in T_{SN}$ in the system net according to the firing rule for high-level Petri nets (e.g., colored Petri nets [13]), as described above.

A synchronization step. Let t be a transition labeled λ in the system net SN, let t be enabled in a marking M w.r.t. a binding b and let $\alpha_1, \ldots, \alpha_k \in \mathrm{A}_{net}$ be net tokens involved in this firing of t. Then t can fire provided that in each α_i ($1 \leqslant i \leqslant k$) a transition labeled by the same synchronization label λ is also enabled. The synchronization step goes then in two stages: first, firing of transitions in all net tokens involved in the firing of t and then, firing of t in the system net w.r.t. binding b.

An NP-net *NP* is called *safe* iff in every reachable marking in *NP* there are not more than one token in each place in the system net, and not more that one token in each net token place. Hereinafter we consider only safe NP-nets.

Now we give a definition of *(strictly) conservative NP-nets*, as well as some related definitions. We then define an unfolding operation for a simple class of strictly conservative nets.

Definition 2. *A safe NP-net* $N = (SN, (EN_1, \ldots, EN_k), \upsilon, \lambda, W)$ *is called* strictly conservative *iff*

1. *For each* $t \in T_{SN}$ *and for each* $p \in {}^\bullet t$, $\exists ! p' \in t^\bullet \,.\, W(p,t) = W(t,p')$ *or* $W(p,t) = \bullet$
2. *For each* $t \in T_{SN}$ *and for each* $p \in t^\bullet$, $\exists ! p' \in {}^\bullet t \,.\, W(p',t) = W(t,p)$ *or* $W(p,t) = \bullet$

The definition of strict conservativeness ensures that no net token emerges or disappears after a transition firing in the system net.

Note that in [4] NP-nets are called conservative, iff tokens cannot disappear after a transition firing, but can be copied; hence, the number of net tokens in such conservative NP-nets can be unlimited. Here we consider a more restrictive subclass of NP-nets with a stable set of net tokens (tokens cannot be copied). Hereinafter we consider only strictly conservative NP-nets, and call them just conservative nets for short.

In conservative nets, instead of considering net tokens (marked element nets residing in places of the system net), we consider *identified* net tokens: triples $\langle I, EN_j, \mu \rangle$, where I is a unique identifier of the token, EN_j is a structure of the token (i.e., an element net from the set $\{EN_1, \ldots EN_k\}$), and μ is a marking of EN_j. Then every net token in the system net has a unique identifier attached to it; thus, tokens with the same marking can be distinguished.

Further we use *NTok* to denote a set of identified net tokens for a given net. Sometimes, by abuse of notation, for a net token $\eta_k = \langle i_k, EN_k, \mu_k \rangle$ in a place x of a marking M, we write $M(x) = \eta_k$ meaning $M(x) = \{(EN_k, \mu_k)\}$. By $\tau(\eta_i)$ we denote a type of a net token (EN_i, μ_i), and by P_{η_i} (T_{η_i}) we denote the set of places (transitions) of the net token, i.e., P_{EN_i} (T_{EN_i}). In the rest of the paper we will use the term *net token* to mean *identified net token.*

Given a system net *SN*, a set of net tokens *NTok*, and a function $\mathfrak{M}$ mapping places of *SN* to identifiers of *NTok*, it is easy to restore the set of element nets (which is just a set of types from *NTok*), and a marking M (which can be easily restored from $\mathfrak{M}$). Thus, we speak about net tokens in a marking as separate entities, and, in order to define an NP-net, we sometimes list identified net tokens.

For a marking M in an NP-net *NP* we define *marking projections* onto the components of *NP*:

1. The projection of M onto a system net *SN*, denoted as $M_{\upharpoonright SN}$, is a marking of the flat P/T-net *SN* obtained by replacing all the net tokens in M by black dot tokens, i.e., $M_{\upharpoonright SN}(p) = |M(p)|$.
2. The projection of M onto a net token $\eta_k = \langle id_k, EN_k, \mu_k \rangle$, denoted as $M_{\upharpoonright \eta_k}$, is just μ_k.

4 Component-Wise Unfoldings

Because in the conservative NP-nets net tokens can be neither created, nor destroyed, it is natural to consider separate unfoldings of the NP-net components. The compositionality of unfoldings can be interesting and helpful for verification. Such unfolding should, of course, possesses some analogue of the the fundamental property of P/T-nets unfoldings.

Definition 3 (Component-wise unfoldings)**.** *A* component-wise *unfolding of an NP-net $NP = (SN, NTok = \{\eta_1, \dots, \eta_k\}, \upsilon, \lambda, W, M^0)$ is an NP-net $U_C(NP) = (U_C(SN), U_C(NTok), U_C(\upsilon), U_C(\lambda), U_C(W), U_C(M^0))$ obtained by separately unfolding each of the components:*

- *$U_C(SN)$ is an unfolding of a system net with the initial marking $M^0_{\restriction SN}$;*
- *$U_C(NTok)$ is a set of net tokens in which every token is unfolded individually.*

Let $h : P_{U_C(SN)} \cup T_{U_C(SN)} \cup_{0<i\leqslant k} P_{U_C(\eta_i)} \cup P_{U_C(\eta_i)} \to P_{SN} \cup T_{SN} \cup_{0<i\leqslant k} P_{\eta_i} \cup P_{\eta_i}$ be a union of all the morphisms from unfoldings to individual components. That is,

$$h(x) = \begin{cases} h_{SN}(x), & \text{if } x \in P_{U_C(SN)} \cup T_{U_C(SN)} \\ h_{\eta_i}(x), & \text{if } x \in P_{U_C(\eta_i)} \cup T_{U_C(\eta_i)} \end{cases}$$

Functions $U_C(W)$,$U_C(\upsilon)$, and $U_C(\lambda)$ are defined in a straightforward manner:

- *$U_C(W)(x, y) = W(h(x), h(y))$;*
- *$U_C(\upsilon)(p) = \upsilon(h(p))$;*
- *$U_C(\lambda)(t) = \lambda(h(t))$.*
- *$U_C(M^0)(p) = M^0(h(p))$ if p is minimal (w.r.t. $<$) in the set $\{p' \mid h(p') = h(p)\}$, or $U_C(M^0)(p) = \varnothing$ otherwise.*

Before addressing the behavioral equivalence of *NP* and $U_C(NP)$, we are going to expand the definition of h to include NP-markings:

Given a marking M_U of $U_C(NP)$, it is possible to define a corresponding marking $h(M_U)$ of *NP*:

$$h(M_U)(p) = \begin{cases} h(S), & \text{if } \exists p' \,.\, h(p') = p \ \wedge \ M_U(p') = S \neq \varnothing \\ \varnothing, & \text{otherwise} \end{cases} \tag{1}$$

The function defined in Eq. (1) is sound because if the first clause of the equation holds, then such p' is unique (because no two elements in the set $\{p' \mid h(p') = x\}$ can be concurrent and, consequently, marked under the same marking).

Theorem 1. *Let NP be a conservative NP-net. Let also M be a reachable marking in NP, and M_U be a reachable marking in $U_C(NP)$ s.t. $h(M_U) = M$.*

1. *If there is a step $M_U \xrightarrow{T_U} M'_U$ in $U_C(NP)$, then there is a step $M \xrightarrow{T} M'$ in NP, such that $h(T_U) = T$ and $h(M'_U) = M'$.*
2. *If there is a step $M \xrightarrow{T} M'$ in NP, then there is a step $M_U \xrightarrow{T_U} M'_U$ in $U_C(NP)$, such that $h(T_U) = T$ and $h(M'_U) = M'$.*

That is, the behavior of an NP-net NP is equivalent to the behavior of its component-wise unfolded counterpart.

The proof is given in the appendix.

While this construction is theoretically apt, it does not give us any direct insight that can be helpful for model checking. For example, while the unfoldings of individual components can be used to analyze behavioral properties for each of the individual components, it is not clear how it can help us to solve verification problems for the entire NP-net. For that reason we apply another approach and construct the occurrence net representing the NP-net semantics directly. The next section is devoted to formalizing the concept of a branching process of a conservative NP-net.

5 Branching Processes of a Conservative NP-net

In this section, we define unfoldings of conservative NP-nets into occurrence nets. We give an inductive definition of a branching process of an NP-net, and (similarly to [5]) define the unfolding as the maximal branching process.

First we introduce a concept of an *element-indexed $\mathcal{C}$-Petri net*, a construction similar to the construction of the canonical net for a P/T-net; however, each place of the element-indexed $\mathcal{C}$-net is paired with a net token (identifier). In this section we suppose that $NP = (SN, (EN_1, \ldots, EN_k), \upsilon, \lambda, M^0)$ is a conservative NP-net, where $SN = (P_{SN}, T_{SN}, F_{SN})$, $EN_i = (P_{EN_i}, T_{EN_i}, F_{EN_i})$, $0 < i \leqslant k$.

Definition 4 (Element-indexed $\mathcal{C}$-Petri nets)**.** *An* element-indexed $\mathcal{C}$-net Θ *for some* element-indexing set J *is a $\mathcal{C}$-net such that each place in Θ is marked with an element of J. For our purposes, the set J will be the set of the identified net tokens.*

Formally, for a fixed net NP, a set of canonical names C is defined as follows:

- *If $x \in (\bigcup_{EN_i} P_{EN_i} \cup P_{SN})$, $\eta_i \in NTok \cup \{\bullet\}$, and X is a finite subset of C, then $(X, x, \eta_i) \in C$;*
- *If $x \subset (\bigcup_{EN_i} T_{EN_i} \cup T_{SN})$, and X is a finite subset of C, then $(X, x) \in C$.*

Then an indexed $\mathcal{C}$-net (P, T, F, M_0) is a P/T-net, such that

1. *$P \cup T \subseteq C$;*
2. *If $p = (X, x, \eta) \in P$, then ${}^\bullet p = X$;*
3. *If $t = (X, x) \in P$, then ${}^\bullet t = X$;*
4. *$(X, x, \eta) \in M_0$ iff $X = \varnothing$ and $x \in (\bigcup_{EN_i} P_{EN_i} \cup P_{SN})$.*

Just like for regular $\mathcal{C}$-Petri nets, there exists a function h mapping the nodes of an element-indexed $\mathcal{C}$-Petri net to the nodes of *NP*:

$$h(x) = \begin{cases} t & \text{if } x = (A, t) \\ p & \text{if } x = (A, p, \eta_i) \end{cases} \tag{2}$$

The union of element-indexed $\mathcal{C}$-Petri nets is defined component-wise, exactly as it was done for regular $\mathcal{C}$-Petri nets.

We also define a notion of an *adjacent place.* According to Definition 2, for every pair $(p, t) \in P_{SN} \times T_{SN}$, where $\upsilon(p) \neq \bullet \wedge ((p, t) \in F_{SN} \vee (t, p) \in F_{SN})$, there exists a unique place p' in a system net such that $W(p, t) = \underbrace{W}(t, p')$ or $W(p', t) = W(t, p)$. Such a place p' is said to be *adjacent to p via t* (denoted by $\langle \widetilde{p, t} \rangle$). For example, in Fig. 5 the place adjacent to p_2 via t_1 is $\langle \widetilde{p_2, t_1} \rangle = p_3$.

Now we are ready to define a set of *element-indexed branching processes* (or *branching processes* for short, when there is no ambiguity) for a given conservative NP-net *NP*.

Definition 5 (Element-indexed branching processes for conservative nested Petri nets)**.** *The set of* element-indexed branching processes *for NP is the smallest set of element-indexed $\mathcal{C}$-nets satisfying the following rules:*

1. *Let*
$$C = \{(\varnothing, p, \eta_i) \mid p \in P_{SN}, \eta_i \in M^0(p)\} \cup \{(\varnothing, p, \eta_i) \mid \eta_i \in NTok, p \in M^0_{\upharpoonright \eta_i}\}$$
be a set of places. The net $\Theta = (C, \varnothing, \varnothing)$ consisting of conditions C and having no transitions is a branching process. Such branching process is said to be initial.

2. *Let Θ be a branching process, and B be a subset of conditions of Θ. If B satisfies the PosEN rule's premise (Fig. 4), then the net obtained by adding an event e and conditions C to Θ is a branching process.*
3. *Let Θ be a branching process, and B be a subset of conditions of Θ. If B satisfies the PosSN rule's premise (Fig. 4), then the net obtained by adding an event e and conditions C to Θ is a branching process.*
4. *Let Θ be a branching process, and let B and B_E be subsets of conditions of Θ. If B and B_E satisfy the PosSync rule's premise (Fig. 4), then the net obtained by adding an event e and conditions C to Θ is a branching process. The SyncCond predicate is defined below.*
5. *Let $\mathcal{BS}$ be a (finite or infinite) set of branching processes. The union $\bigcup\mathcal{BS}$ is a branching process.*

In rules (2)-(4), event e is called a possible extension *of Θ.*

$$\frac{B=\{(x_i,b_i,\eta_k)\mid i\in I\}\qquad co(B)\qquad t\in T_{\eta_k},\ \lambda(t)=\bot\qquad {}^\bullet t=\{b_i\mid i\in I\}}{e=(B,\{t\})\quad\text{and}\quad C=\bigcup_{p\in t^\bullet}(e,p,\eta_k)}\ PosEN$$

$$\begin{array}{l}B=\{(x_i,b_i,\eta_i)\mid i\in I\}\\ \dfrac{co(B)\qquad\qquad t\in T_{SN},\ \lambda(t)=\bot\qquad {}^\bullet t=\{b_i\mid i\in I\}}{\begin{array}{l}e=(B,\{t\})\quad\text{and}\quad C=\{(e,\widetilde{\langle b_i,t\rangle},\eta_i)\mid i\in I,\eta_i\neq\bullet\}\cup\\ \qquad\qquad\qquad\qquad\qquad\{(e,b,\bullet)\mid b\in t^\bullet,\upsilon(b)=\bullet\}\end{array}}\ PosSN\end{array}$$

$$\begin{array}{l}B=\{(x_i,b_i,\eta_i)\mid i\in I\}\qquad t\in T_{SN},\ \lambda(t)\neq\bot\\ \dfrac{co(B\cup B_E)\qquad {}^\bullet t=\{b_i\mid i\in I\}\qquad SyncCond(B_E,E,I,\Theta,(B,t))}{\begin{array}{l}e=(B\cup B_E,\{t\}\cup E)\quad\text{and}\quad C=\{(e,\widetilde{\langle b_i,t\rangle},\eta_i)\mid i\in I,\eta_i\neq\bullet\}\cup\\ \qquad\qquad\qquad\qquad\qquad\{(e,b,\bullet)\mid b\in t^\bullet,\upsilon(b)=\bullet\}\cup\\ \qquad\qquad\qquad\qquad\qquad\{(t',c_i',\eta_i)\mid i\in I,\eta_i\neq\bullet,c_i'\in P_{\eta_i},c_i'\in {t_i}^\bullet\}\end{array}}\ PosSync\end{array}$$

Figure 4: Rules for possible extensions of a branching process

The *SyncCond* predicate in rule (4) makes sure that all the components involved in the synchronization step, synchronize correctly. The parameter I contains the id's of all the net tokens involved in the step. The set E consists of transitions t_i in each of the net tokens η_i ($i\in I$), and every t_i carries the same label as the transition t from the system net. In order for the synchronization step to go through, each of the t_i needs to have its pre-set $\{c_j\mid j\in J_i\}$ active. The places of net tokens corresponding to those in the pre-sets are contained in B_E.

We say that the $SyncCond(B_E,E,I,\Theta,(B,t))$ predicate is true iff the following conditions hold:

1. $B_E=\bigcup_{i\in I}\{(y_j,c_j,\eta_i)\mid j\in J_i,\eta_i=(id_i,EN_i,\mu_i)\in NTok,c_j\in P_{EN_i}\}\wedge co(B_E)$, i.e., B_E is a set of reachable conditions that correspond to places in net tokens;

2. $E = \{t_i \in T_{EN_i} \mid i \in I, \eta_i = (id_i, EN_i, \mu_i) \in NTok\}$, i.e., E is a subset of transitions in each of the net tokens;
3. $\forall t_i \in E$, $\lambda(t_i) = \lambda(t)$
4. ${}^\bullet E = \bigcup_{i \in I} \{c_j \mid j \in J_i, \eta_i \in NTok\}$

The rules in Fig. 4 can be explained informally from the operational point of view. Rules $PosEN$, $PosSN$, and $PosSync$ are used for generating events that correspond to element-autonomous, system-autonomous, and synchronized firings, respectively.

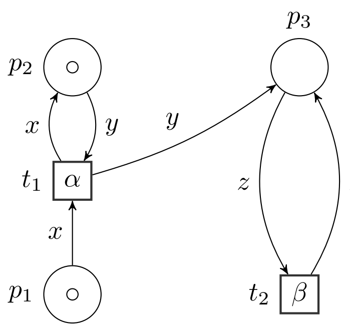

Figure 5: NP-net NP_2: System net

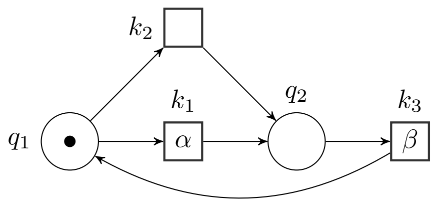

Figure 6: NP-net NP_2: Element net

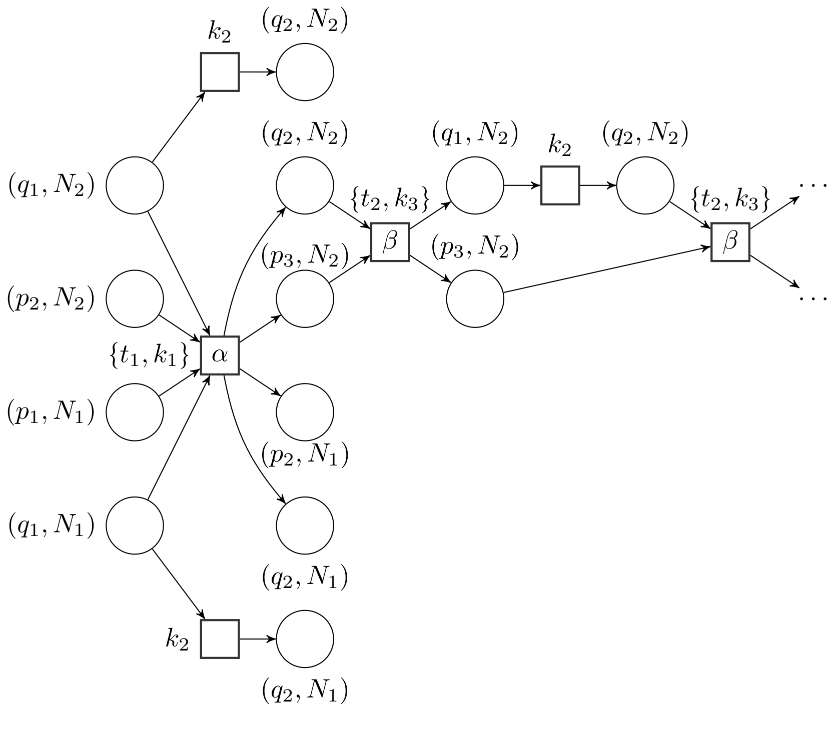

Figure 7: Branching process of NP_2

For example, consider NP-net NP_2 depicted in Figs. 5 and 6. A possible branching process of NP_2 is shown in Fig. 7. In Fig. 7, a transition is labeled with t, if it is of the form (A, t), and a place is labeled with (p, N) if it is of the form (A, p, N).

Properties of element-indexed branching processes. Below we formulate some important properties of the constructs that we have defined.

Proposition 1. *Every element-indexed branching process is an occurrence net.*

The proof is given in the appendix.

Let B be an element-indexed branching process of *NP*. Next, we define the function $\tilde{h}$, which is an extension of the labeling function h from Eq. (2). The function $\tilde{h}$ maps the markings of B to the markings of *NP*. In the case of flat P/T-nets, such an ad-hoc extension is not necessary, as a marking of a flat P/T-net is simply a multiset over a set of places, and the value of h on a multiset is defined in a usual manner. In the case of NP-nets, however, a marking is a function mapping system net places to multisets of net tokens. It should be noted that in a case when the NP-net does not contain net tokens, the definition of $\tilde{h}$ coincides with that of h.

$$\tilde{h}(M)(p) = \begin{cases} \{\bullet\} & \text{if } \upsilon(p) = \bullet \text{ and } h(M)(p) > 0 \\ \{\{q \mid q \in P_{\eta_i}, \exists A, (A, q, \eta_i) \in M\}\} & \text{if } \upsilon(p) = \tau(\eta_i) \text{ and } \exists A, (A, p, \eta_i) \in M \end{cases} \tag{3}$$

The mapping is sound because for every net token η_k in every reachable marking of a branching process, only one condition labeled with η_k can be marked. Because every branching process is an occurrence net, the reachable conditions of a branching process correspond to the pairwise concurrent sets. Therefore, we have to prove the following lemma:

Lemma 1. *Let η_k be a net token and θ be a branching process. If $p_i, p_j \in P_{SN}$, $p_i \neq p_j$ and $(A, p_i, \eta_k), (B, p_j, \eta_k) \in \theta$, then $\neg((A, p_i, \eta_k)\, co\, (B, p_j, \eta_k))$.*

The proof is given in the appendix.

It should be noted that every low-level P/T-net is a special case of an NP-net with the empty set of element nets and no vertical synchronization. The following property is a "sanity check" as it ensures that our branching process definition is in accord with the branching process definition for low-level Petri nets.

Proposition 2. *Let N be a P/T-net. The set of branching processes of N is isomorphic to the set of element-indexed branching processes of N, when N is considered as an NP-net.*

Proof. The only possible extension rule, that can be applied in the case of P/T-net is *PosSN*. Furthermore, in this case, the *PosSN* rule coincides with the possible extension rule of the low-level branching processes. □

We also extend the definition of the prefix relation $\sqsubseteq$ to cover the branching processes of nested Petri nets. Then, the *unfolding* of a conservative nested Petri net *NP* (denoted by $U(NP)$) is the $\sqsubseteq$-maximal branching process of *NP*.

Now it is possible to see that the fundamental property of unfoldings holds for our definition of the conservative NP-nets unfolding.

Theorem 2. *Let M be a reachable marking in a nested Petri net NP and M_U be a reachable marking in $U(NP)$ s.t. $\tilde{h}(M_U) = M$.*

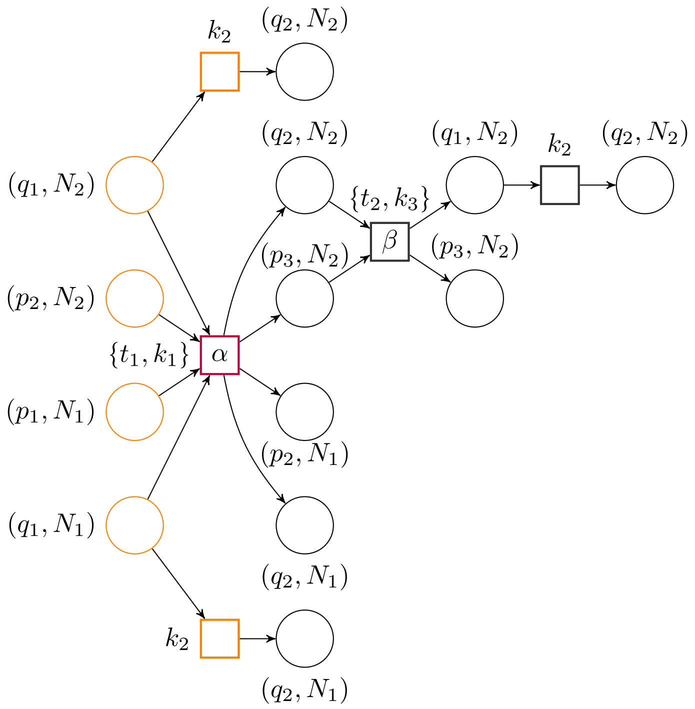

Figure 8: Complete branching process BP_c of NP_2

1. *If there is a step* $M_U \xrightarrow{t_U} M'_U$ *in* $U(NP)$*, then there is a step* $M \xrightarrow{t} M'$ *in* NP*, such that* $\tilde{h}(M'_U) = M'$ *and* $h(t_U) = t$*.*
2. *If there is a step* $M \xrightarrow{t} M'$ *in* NP*, then there is a step* $M_U \xrightarrow{t_U} M'_U$ *in* $U(NP)$*, such that* $\tilde{h}(M'_U) = M'$ *and* $h(t_U) = t$*.*

Application to verification. Thanks to the results outlined in this section, the basic algorithm (described in [14]) for constructing finite prefixes of unfoldings of low-level P/T-nets can be modified in a straightforward way to obtain an algorithm for constructing finite prefixes of unfoldings of conservative NP-nets. In fact, the only part of the algorithm that needs to be modified is the PotExt function, which has to be changed in accordance with the possible extension rules in Fig. 4. This is attainable because all the necessary definitions (in particular, the definition of a *cutting context*) and the theory of canonical prefixes [16, 14] can be directly extended to cover NP-nets.

For example, let's consider the result of the standard algorithm applied to the NP-net NP_2 from Figs. 5 and 6 using the McMillan's cutting context (in the notation of [14], $C' \approx C'' \iff Mark(C') = Mark(C'')$ и $C' \lhd C'' \iff |C'| < |C''|$). The resulting canonical prefix can be seen in Fig. 8.

This canonical prefix BP_C allows us to solve the executability problem: a transition t may fire in the NP-net iff an event labeled with t is present in the canonical branching process. For example, one can observe, that because a transition e_1 (colored red) in BP_2 has is labeled with $\{t_1, k_1\}$, the transition t_1 is executable in the NP-net.

One can also use the canonical prefix BP_C to check the net for deadlocks. A deadlock is present in the NP-net iff there exists a configuration in the prefix BP_C that does not contain cut-off events and lead to a deadlock marking in BP_C. For example, in Fig. 8 such configuration is colored orange. This configuration corresponds to a marking in the NP-net, in which both of the net tokens are located in position p_1 and p_2 in the system net, and both net tokens has the same marking $\{q_2\}$. It is obvious that such a marking is a deadlock.

6 Conclusion

In this paper, we have defined branching processes of conservative nested Petri nets. An unfolding of an NP-net is defined as the maximal branching processes. It is shown that the unfolding posses the same behavior as the original net. Our definition of branching processes allows us to reuse most of the techniques used for the verification via the canonical prefixes of net unfoldings. An algorithm for generating finite canonical prefixes of unfoldings differs from the standard one only in the PotExt function, which is used for generating possible extensions of branching processes. We have also presented a component wise unfolding construction and have proven that it correlates with the NP-net definition.

In future work, we plan to examine a possibility of extending this technique to other NP-net classes.

References

[1] Paolo Baldan, Andrea Corradini, Barbara König, and Stefan Schwoon. Mcmillan's complete prefix for contextual nets. In Kurt Jensen, Will M.P. Aalst, and Jonathan Billington, editors, *Transactions on Petri Nets and Other Models of Concurrency I*, volume 5100 of *Lecture Notes in Computer Science*, pages 199–220. Springer, 2008.

[2] Marek A. Bednarczyk, Luca Bernardinello, Wiesław Pawłowski, and Lucia Pomello. Modelling mobility with petri hypernets. In *Recent Trends in Algebraic Development Techniques*, volume 3423 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2005.

[3] Nina Buchina and Leonid Dworzański. The tool for modeling of wireless sensor networks with nested Petri nets. In *Proceedings of 7th the Spring/Summer Young Researchers' Colloquium on Software Engineering*, SYRCoSE '13, pages 15–18. Institute for System Programming of the Russian Academy of Sciences, 2013.

[4] Leonid W. Dworzański and Irina A. Lomazova. On compositionality of boundedness and liveness for nested Petri nets. *Fundamenta Informaticae*, 120(3):275–293, 2012.

[5] Joost Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28(6):575–591, 1991.

[6] Javier Esparza and Keijo Heljanko. *Unfoldings: a partial-order approach to model checking.* Springer, 2008.

[7] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 87–106. Springer, 1996.

[8] Berndt Farwer and Irina Lomazova. A systematic approach towards object-based Petri net formalisms. In Dines Bjørner, Manfred Broy, and AlexandreV. Zamulin, editors, *Perspectives of System Informatics*, volume 2244 of *Lecture Notes in Computer Science*, pages 255–267. Springer, 2001.

[9] Hans Fleischhack and Christian Stehno. Computing a finite prefix of a time Petri net. In Javier Esparza and Charles Lakos, editors, *Application and Theory of Petri Nets 2002*, volume 2360 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2002.

[10] Kees M. van Hee, Irina A. Lomazova, Olivia Oanea, Alexander Serebrenik, Natalia Sidorova, and Marc Voorhoeve. Nested nets for adaptive systems. In *Petri Nets and Other Models of Concurrency-ICATPN 2006*, pages 241–260. Springer, 2006.

[11] Keijo Heljanko. Deadlock and reachability checking with finite complete prefixes. Technical report, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 1999.

[12] Kathrin Hoffmann, Hartmut Ehrig, and Till Mossakowski. High-level nets with nets and rules as tokens. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2005.

[13] Kurt Jensen and Lars M. Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer, 2009.

[14] Victor Khomenko. *Model Checking Based on Prefixes of Petri Net Unfoldings*. Ph.D. Thesis, School of Computing Science, Newcastle University, 2003.

[15] Victor Khomenko and Maciej Koutny. Branching processes of high-level Petri nets. In Hubert Garavel and John Hatcliff, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2619 of *Lecture Notes in Computer Science*, pages 458–472. Springer, 2003.

[16] Victor Khomenko, Maciej Koutny, and Walter Vogler. Canonical prefixes of Petri net unfoldings. *Acta Informatica*, 40(2):95–118, 2003.

[17] Michael Köhler-Bußmeier and Frank Heitmann. Conservative elementary object systems. *Fundamenta Informaticae*, 120(3):325–339, 2012.

[18] Michael Köhler and Berndt Farwer. Object nets for mobility. In Jetty Kleijn and Alex Yakovlev, editors, *Petri Nets and Other Models of Concurrency – ICATPN 2007*, volume 4546 of *Lecture Notes in Computer Science*, pages 244–262. Springer, 2007.

[19] Rom Langerak and Ed Brinksma. A complete finite prefix for process algebra. In Nicolas Halbwachs and Doron Peled, editors, *Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 184–195. Springer, 1999.

[20] Irina A. Lomazova. Nested Petri nets—a formalism for specification and verification of multi-agent distributed systems. *Fundamenta Informaticae*, 43(1):195–214, 2000.

[21] Irina A. Lomazova. Modeling dynamic objects in distributed systems with nested Petri nets. *Fundamenta Informaticae*, 51(1-2):121–133, 2002.

[22] Irina A. Lomazova. *Vlozhennye seti Petri: modelirovanie i analiz raspredelennykh sistem s ob"ektnoy strukturoy (in Russian).* Nauchnyy Mir, Moscow, 2004.

[23] Irina A. Lomazova. Nested Petri nets for adaptive process modeling. In *Pillars of computer science*, pages 460–474. Springer, 2008.

[24] Marco Mascheroni. *Hypernets: a Class of Hierarchical Petri Nets.* Ph.D. Thesis, Facolt di Scienze Naturali Fisiche e Naturali, Dipartimento di Informatica Sistemistica e Comunicazione, Università Degli Studi Di Milano Bicocca, 2010.

[25] Kenneth L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In Gregor Bochmann and DavidKarl Probst, editors, *Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 1993.

[26] Kenneth L. McMillan. A technique of state space search based on unfolding. *Form. Methods Syst. Des.*, 6(1):45–65, 1995.

[27] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

[28] Rüdiger Valk. Object Petri nets. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer, 2004.

A Proof of Theorem 1

Proof. We provide a proof for the first part of the theorem, and consider only the synchronization step. Other cases can be proved in a similar way.

For each of the components $U_C(N_i), i \in I_T$ involved in the firing of the T_U, there is a transition $t_i \in T_U$. Additionally, there is $t_S \in T_U$ that belongs to the system net.

For each t_i there exists p_i in which the net token $U_C(N_i)$ resides. It is then the case that ${}^\bullet t_i \subseteq M_U(p_i)$. Furthermore, if the firing of T moves the token $U_C(N_i)$ from p_i to some p'_i, then we have $M'_U(p'_i) = M_U(p_i) \backslash {}^\bullet t_i \cup t_i{}^\bullet$. This means that in a net token (viewed as a separate component), $M_U(p_i) \xrightarrow{t_i} M'_U(p'_i)$. But then, according to the fundamental property of unfoldings, $h(M_U(p_i)) = M(h(p_i)) \xrightarrow{h(t_i)} M'(h(p'_i)) = h(M'_U(p'_i))$ in N_i.

Further, there exists a step $M_U \xrightarrow{t_S} M'_U$ in $U_C(SN)$ viewed as a separate component. Hence, there exists a step $h(M_{U\restriction SN}) \xrightarrow{h(t_S)} h(M'_{U\restriction SN})$ in SN. Combined with the fact that U_C preserves labels on transitions, this implies that each transition in $T = h(T_U)$ is active in NP under M. Moreover, it is easy to see that M' coincides with $h(M'_U)$. □

B Proof of Proposition 1

Proof. Let $\Theta = (P_\Theta, T_\Theta, F_\Theta)$ be an element-indexed branching process. We prove that Θ is an occurrence net, i.e., that all four conditions from the definition of the occurrence net hold.

1. This condition holds simply because Θ is an element-indexed $\mathcal{C}$-Petri net.

2. We can verify that this prerequisite holds by noting that every rule makes sure that each place has at most one pre-transition (keeping in mind the way that pre-sets are defined for element-indexed $\mathcal{C}$-nets).
3. Validity of this conditions follows immediately from the inductive rules.
4. Assume that there exists a self-conflicting node in the branching process. Let $x \in P_\Theta \cup T_\Theta$ be the minimal self conflicting node ($x\#x$). That means that $\exists t, t' \in T_\Theta, t \neq t'.\ \ t < x \wedge t' < x \wedge {}^\bullet t \cap {}^\bullet t' \neq \varnothing$.

   If x is a place, then there exists a unique (according to the case 2) transition y such that ${}^\bullet x = \{y\}$ and $y\#y$. Furthermore, $x\#x \iff y\#y$; therefore, we are to consider only the case of x being an event.

   This means that x has a form of (B, t), and there exist conditions $b, b' \in B$ such that $b\#b'$, i.e., $\neg co(B)$, but this is is impossible, according to the rules.

□

C Proof of Lemma 1

Proof. This can be proven by the structural induction on the definition of a branching process. The statement clearly holds for the initial branching process.

Assume that the lemma holds for θ and θ' is an extension of θ obtained by applying one of the possible extension rules. Let C be the set of newly added conditions labeled with places from the system net. One can notice that all conditions in C are labeled with a different net tokens. Therefore, in order to prove the lemma we only are to show that $\forall (A, p_i, \eta_k) \in \theta$, $\forall (B, p_j, \eta_k) \in C$, $\neg((A, p_i, \eta_k)\ co\ (B, p_j, \eta_k))$.

The $PosEN$ rule does not introduce new conditions labeled with places from the system net, so the set C is empty. As for the $PosSN$ rule, it is possible to notice that there exists a condition (A', p_l, η_k) from θ, such that $(A', p_l, \eta_k) \in {}^{\bullet\bullet}(B, p_j, \eta_k)$ (where p_j is adjacent to p_l). Due to the inductive hypothesis, $\neg((A', p_l, \eta_k)\ co\ (A, p_i, \eta_k))$. Therefore, $\neg((A, p_i, \eta_k)\ co\ (B, p_j, \eta_k))$.

The $PosSync$ case can be handled in a similar way.

□